```
/*
Display.cpp - A simple track GPS to SD card logger. Display module.
TinyTrackGPS v0.6

Copyright © 2019-2021 Francisco Rafael Reyes Carmona.
All rights reserved.

rafael.reyes.carmona@gmail.com

  This file is part of TinyTrackGPS.

  TinyTrackGPS is free software: you can redistribute it and/or modify
  it under the terms of the GNU General Public License as published by
  the Free Software Foundation, either version 3 of the License, or
  (at your option) any later version.

  TinyTrackGPS is distributed in the hope that it will be useful,
  but WITHOUT ANY WARRANTY; without even the implied warranty of
  MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
  GNU General Public License for more details.

  You should have received a copy of the GNU General Public License
  along with TinyTrackGPS.  If not, see <https://www.gnu.org/licenses/>.
*/

#include "Display.h"

Display::Display(Display_Type t):_screen(t){
    if (_screen == SDD1306_128X64){
        _width = 16;
        _height = 8;
        _offset = 0;
    } else if (_screen == LCD_16X2 || _screen == LCD_16X2_I2C){
        _width = 16;
        _height = 2;
        _offset = 0;
    }
}

void Display::start(){
    if (_screen == LCD_16X2 || _screen == LCD_16X2_I2C){
        #if defined(DISPLAY_TYPE_LCD_16X2) || defined(DISPLAY_TYPE_LCD_16X2_I2C)
        // DEFINICION DE CARACTERES PERSONALIZADOS
        byte alt[8] = {
            0b00000100,
            0b00001110,
            0b00011111,
            0b00000100,
            0b00000100,
            0b00000100,
            0b00000100,
            0b00000100,
        };

        byte ant[8] = {
            0b00001110,
            0b00010001,
            0b00010101,
            0b00010001,
            0b00000100,
```

```
            0b00000100,
            0b00001110,
            0b00000000,
        };

        byte sd[8] = {
            0b00001110,
            0b00010001,
            0b00011111,
            0b00000000,
            0b00000000,
            0b00010111,
            0b00010101,
            0b00011101,
        };

        byte hourglass_0[8] = {
            0b00011111,
            0b00001110,
            0b00001110,
            0b00000100,
            0b00000100,
            0b00001010,
            0b00001010,
            0b00011111,
        };

        byte hourglass_1[8] = {
            0b00011111,
            0b00001010,
            0b00001110,
            0b00000100,
            0b00000100,
            0b00001010,
            0b00001010,
            0b00011111,
        };

        byte hourglass_2[8] = {
            0b00011111,
            0b00001010,
            0b00001110,
            0b00000100,
            0b00000100,
            0b00001010,
            0b00001110,
            0b00011111,
        };

        byte hourglass_3[8] = {
            0b00011111,
            0b00001010,
            0b00001010,
            0b00000100,
            0b00000100,
            0b00001010,
            0b00001110,
            0b00011111,
        };

```

```
        byte hourglass_4[8] = {
            0b00011111,
            0b00001010,
            0b00001010,
            0b00000100,
            0b00000100,
            0b00001110,
            0b00001110,
            0b00011111,
        };
        #endif
        #if defined(DISPLAY_TYPE_LCD_16X2)
        lcd = new LiquidCrystal(RS, ENABLE, D0, D1, D2, D3);
        lcd->begin(_width, _height);
        #elif defined(DISPLAY_TYPE_LCD_16X2_I2C)
        lcd = new LiquidCrystal_I2C(I2C,_width,_height);
        lcd->init();
        lcd->backlight();
        #endif

        #if defined(DISPLAY_TYPE_LCD_16X2) || defined(DISPLAY_TYPE_LCD_16X2_I2C)
        lcd->createChar(0, hourglass_0);
        lcd->createChar(1, hourglass_1);
        lcd->createChar(2, hourglass_2);
        lcd->createChar(3, hourglass_3);
        lcd->createChar(4, hourglass_4);
        lcd->createChar(5, alt);
        lcd->createChar(6, ant);
        lcd->createChar(7, sd);
        #endif
    }

    if (_screen == SDD1306_128X64) {
        #if defined(DISPLAY_TYPE_SDD1306_128X64)
        u8x8_SSD1306 = new U8X8_SSD1306_128X64_NONAME_HW_I2C(U8X8_PIN_NONE, SCL,
SDA);
        u8x8_SSD1306->begin();
        u8x8_SSD1306->setFont(u8x8_font_7x14B_1x2_r);
        #endif
    }
}

void Display::clr(){
    if (_screen == LCD_16X2 || _screen == LCD_16X2_I2C) {
        #if defined(DISPLAY_TYPE_LCD_16X2) || defined(DISPLAY_TYPE_LCD_16X2_I2C)
        lcd->clear();
        #endif
    }
    else if (_screen == SDD1306_128X64) {
        #if defined(DISPLAY_TYPE_SDD1306_128X64)
        u8x8_SSD1306->clear();
        #endif
    }
}

void Display::print(int vertical, int horizontal, const char text[]){
    if (_screen == LCD_16X2 || _screen == LCD_16X2_I2C) {
        #if defined(DISPLAY_TYPE_LCD_16X2) || defined(DISPLAY_TYPE_LCD_16X2_I2C)
        lcd->setCursor(vertical, horizontal);
        lcd->print(text);
```

```
        #endif
    }
    else if (_screen == SDD1306_128X64) {
        #if defined(DISPLAY_TYPE_SDD1306_128X64)
        //u8x8_SSD1306->setCursor(vertical, (horizontal*2));
        //u8x8_SSD1306->print(text);
        u8x8_SSD1306->setCursor(vertical, (horizontal*2));
        this->print(text);
        //u8x8_SSD1306->display();
        #endif
    }
}

void Display::print(int line, const char text[]){
    byte pos = _width -(strlen(text));
    pos = (pos >> 1);
    this->print((int)pos, line, text);
}

void Display::print(const char text[]){
    if (_screen == LCD_16X2 || _screen == LCD_16X2_I2C) {
    #if defined(DISPLAY_TYPE_LCD_16X2) || defined(DISPLAY_TYPE_LCD_16X2_I2C)
    lcd->print(text);
    #endif
    }
    else if (_screen == SDD1306_128X64) {
        #if defined(DISPLAY_TYPE_SDD1306_128X64)
        u8x8_SSD1306->print(text);
        u8x8_SSD1306->flush();
        #endif
    }
}

void Display::print(const char text1[], const char text2[]){
    if (_screen == LCD_16X2 || _screen == LCD_16X2_I2C) {
        this->print(0, text1);
        this->print(1, text2);
    }
    else if (_screen == SDD1306_128X64) {
        this->print(1, text1);
        this->print(2, text2);
    }
}

void Display::print(const char text1[], const char text2[], const char text3[]){

}

void Display::print(const char text1[], const char text2[], const char text3[],
const char text4[]){

}

void Display::wait_anin(unsigned int t){
    if (_screen == LCD_16X2 || _screen == LCD_16X2_I2C) {
        #if defined(DISPLAY_TYPE_LCD_16X2) || defined(DISPLAY_TYPE_LCD_16X2_I2C)
        lcd->setCursor(15,1);
        lcd->write((byte)t%5);
        #endif
    }
```

```
    else if (_screen == SDD1306_128X64) {
        #if defined(DISPLAY_TYPE_SDD1306_128X64)
        //char p = 0x2c;
        //u8x8_SSD1306->drawString((t%16),6,"-");

        uint8_t hourglass_UP[5][8] = {  0x01,0x1f,0x7f,0xff,0xff,0x7f,0x1f,0x01,
                                        0x01,0x1f,0x7d,0xf9,0xf9,0x7d,0x1f,0x01,
                                        0x01,0x1f,0x79,0xf1,0xf1,0x79,0x1f,0x01,
                                        0x01,0x1f,0x71,0xe1,0xe1,0x71,0x1f,0x01,
                                        0x01,0x1f,0x61,0x81,0x81,0x61,0x1f,0x01
                                        };

        uint8_t hourglass_DOWN[5][8] = {0x80,0xf8,0x86,0x81,0x81,0x86,0xf8,0x80,
                                        0x80,0xf8,0xc6,0xe1,0xe1,0xc6,0xf8,0x80,
                                        0x80,0xf8,0xe6,0xf1,0xf1,0xe6,0xf8,0x80,
                                        0x80,0xf8,0xfe,0xf9,0xf9,0xfe,0xf8,0x80,
                                        0x80,0xf8,0xfe,0xff,0xff,0xfe,0xf8,0x80
                                        };
        u8x8_SSD1306->drawTile((_width>>1)-1, 6, 1, hourglass_UP[t%5]);
        u8x8_SSD1306->drawTile((_width>>1)-1, 7, 1, hourglass_DOWN[t%5]);
        #endif
    }
}

void Display::print_PChar(byte c) {
    if (_screen == LCD_16X2 || _screen == LCD_16X2_I2C) {
    #if defined(DISPLAY_TYPE_LCD_16X2) || defined(DISPLAY_TYPE_LCD_16X2_I2C)
    lcd->write(c);
    #endif
    }
    else if (_screen == SDD1306_128X64) {
    #if defined(DISPLAY_TYPE_SDD1306_128X64)
    uint8_t PChar_UP[3][8] =      { 0x30,0x38,0x3c,0xff,0xff,0x3c,0x38,0x30,
                                    0x3c,0x02,0x01,0xd9,0xd9,0x01,0x02,0x3c,
                                    0x78,0x7c,0x6e,0x66,0x66,0x6e,0x7c,0x78
                                    };
    uint8_t PChar_DOWN[3][8] =    { 0x00,0x00,0x00,0xff,0xff,0x00,0x00,0x00,
                                    0x00,0xc0,0xe0,0xff,0xff,0xe0,0xc0,0x00,
                                    0x7c,0xfc,0xc0,0xf8,0x7c,0x0c,0xfc,0xf8
                                    };
    if (c == 5) {
        u8x8_SSD1306->drawTile(9, 2, 1, PChar_UP[0]);
        u8x8_SSD1306->drawTile(9, 3, 1, PChar_DOWN[0]);
    }
    else if (c == 6) {
        u8x8_SSD1306->drawTile(11, 0, 1, PChar_UP[1]);
        u8x8_SSD1306->drawTile(11, 1, 1, PChar_DOWN[1]);
    }
    else if (c == 7) {
        u8x8_SSD1306->drawTile(15, 0, 1, PChar_UP[2]);
        u8x8_SSD1306->drawTile(15, 1, 1, PChar_DOWN[2]);
    }
    #endif
    }
}

void Display::splash(int time_delay){
    this->print(NAME, VERSION);
    delay(time_delay);
}
```